


Ejercicio Práctico 1

 **Tema:** Detección y mitigación de múltiples vulnerabilidades web

Objetivo del ejercicio:

Analizar un pequeño sistema vulnerable para identificar y mitigar amenazas de tipo **SQL Injection**, **Cross-Site Scripting (XSS)** y **Cross-Site Request Forgery (CSRF)** utilizando buenas prácticas y herramientas básicas de seguridad.

Escenario:



Te han entregado un fragmento de código de un sistema de comentarios desarrollado en PHP. Debes identificar posibles vulnerabilidades y aplicar medidas correctivas.

```
<?php
$nombre = $_POST['nombre'];
$comentario = $_POST['comentario'];

$query = "INSERT INTO comentarios (nombre, comentario) VALUES ('$nombre',
'$comentario')";
mysqli_query($conexion, $query);
?>
```

Parte 1: Análisis de vulnerabilidades

a. ¿Qué vulnerabilidades puedes identificar en este código? Marca todas las que correspondan.

- Inyección SQL 
- XSS 

- CSRF
 - Fuerza Bruta
-

b. ¿Qué ataque XSS podrías ejecutar en el campo `comentario`?

- `<script>alert('XSS');</script>` ✓
 - `DROP TABLE comentarios;`
 - `<meta charset="utf-8">`
-

c. ¿Cómo evitarías el XSS en este sistema?

- No usar HTML
 - Usar JavaScript para filtrar
 - Escapar caracteres especiales con `htmlspecialchars()` ✓
-

Parte 2: Corrección del código

Reescribe el código para:

- Evitar la inyección SQL utilizando **prepared statements**
 - Prevenir XSS escapando la salida
 - Incluir un **token CSRF** en el formulario (parte cliente + parte servidor)
-

Solución esperada (resumen simplificado):

Formulario HTML:

```
<form method="POST" action="comentarios.php">
  <input type="text" name="nombre">
```

```
<textarea name="comentario"></textarea>
<input type="hidden" name="csrf_token" value="<?php echo $_SESSION['csrf_token'];
?>">
<input type="submit" value="Enviar">
</form>
```

Código PHP seguro:

```
<?php
session_start();
if ($_POST['csrf_token'] !== $_SESSION['csrf_token']) {
    die("CSRF Token inválido");
}

$nombre = htmlspecialchars($_POST['nombre'], ENT_QUOTES, 'UTF-8');
$comentario = htmlspecialchars($_POST['comentario'], ENT_QUOTES, 'UTF-8');

$stmt = $conexion->prepare("INSERT INTO comentarios (nombre, comentario) VALUES (?,
?);");
$stmt->bind_param("ss", $nombre, $comentario);
$stmt->execute();
?>
```



Parte 3: Evaluación práctica con herramienta

Instrucciones:

1. Ejecuta OWASP ZAP o Burp Suite en un entorno controlado.
 2. Accede al formulario vulnerable original (antes de aplicar la solución).
 3. Identifica los parámetros vulnerables a SQLi y XSS.
 4. Anota al menos **2 hallazgos** y describe cómo fueron corregidos en tu versión segura.
-



Evaluación: Se considerará aprobado el ejercicio si el estudiante:

- Identifica correctamente las vulnerabilidades.
- Aplica las tres técnicas de mitigación (SQLi, XSS, CSRF).

- Utiliza alguna herramienta para validar sus correcciones.

